ESC-TR-94-114

Project Report
RASSP-2

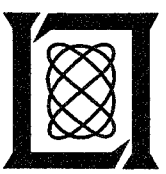
RASSP Benchmark 1 Executable Requirements User's Manual

A.H. Anderson
G.A. Shaw

20 December 1994

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

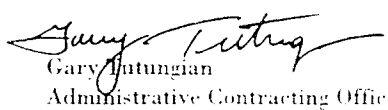


Prepared for the Advanced Research Projects Agency
under Air Force Contract F19628-95-C-0002.

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. The work was sponsored by the Advanced Research Projects Agency under Air Force Contract F19628-95-C-0002.

This project report has been reviewed and is approved for publication.

FOR THE COMMANDER



Gary Tutungian
Administrative Contracting Officer
Contracted Support Management

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

RASSP BENCHMARK 1 EXECUTABLE REQUIREMENTS USER'S MANUAL

A.H. ANDERSON
Group 23

G.A. SHAW
Group 97

PROJECT REPORT RASSP-2

20 DECEMBER 1994

LEXINGTON MASSACHUSETTS

ABSTRACT

This User's Manual describes the installation and use of a set of computer programs and data files which comprise the Executable Requirement for the first benchmark in the RASSP program. The VHDL programs provide a simulation of a Synthetic Aperture Radar processor and a test bench for the simulation. Several C-language utility programs and an image display program are provided. Two sets of data, one from an airborne radar and one synthetic, and images created from the data by a C-language implementation of the SAR algorithm are described. Example scripts for execution of the simulation on Vantage and Mentor simulators are presented along with performance data from example runs.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. Introduction

The RASSP Executable Requirements is a part of RASSP Benchmark 1, described in *RASSP Benchmark-1, Synthetic Aperture Radar Image Processor.* [1] This document, hereafter referred to as BM1, is essential to understanding this User's Manual. The Executable Requirements comprises a VHDL model of the processor and a test bench as shown in Figure 1, input data files, comparison output image files, and other files and programs as described in this document. The processor model implements the SAR image formation algorithm described in BM1. The test bench sources and sinks data for the processor, supplies commands and setup data on the control interface, checks latency of the processor and compares output data with data in a comparison file.

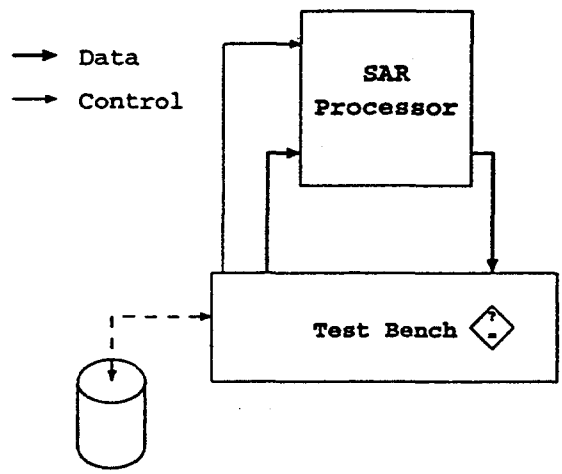


Figure 1. VHDL implementation of SAR processor and test bench.

[1] RASSP Benchmark-1, Synthetic Aperture Radar Image Processor, MIT Lincoln Laboratory, Lexington, MA, July, 1994. Available on request.

2. Processor Model

The following sections describe the fidelity of the processor model to the system description in BM1.

2.1 SAR Processor

The processor model is an algorithmic model which is identical to the description in BM1. It is implemented in VHDL with the VHDL `real` type. In the Vantage simulator the arithmetic is performed in double precision. The only timing of the processor model is at the input and output ports and the latency from input to output. Latency of the processor is controlled by the `latency` generic described in Table 3. No claim is made as to the algorithmic efficiency of this VHDL model.

2.2 Data Ports

Both the input and output data ports are modeled as if the interface between the processor and test bench is the 40 bit parallel interface of the TriQuint HRC-500FS module described in BM1 Appendix C. There is no modeling of the optical transmission system. Data is input to the processor at a rate of 4.56 MW/s synchronized with a 12.5 MHz clock. Output from the processor is at a rate of 8.0 MW/s synchronized with a 12.5 MHz clock.

2.3 Control and Diagnostic Port

The C/D port is different from the description in BM1. It is a 32 bit unidirectional port used only for delivering commands and setup data from the test bench to the processor. The subset of the BM1 commands which are implemented are shown in Table 1. Two one-bit synchronizing signals are used on the interface; `diagstrobe` indicates the presence of a new word on the interface and `loadcmd` indicates that the word is a command word rather than data. On the 32 bit port commands are encoded in the four least significant bits as shown in the **Code** field of Table 1.

TABLE 1

Control/Diagnostic Commands

Command	Number of data lines	Data type	Function	Code
reboot	–	–	Reboot the processor	0000
restart	–	–	Restart the processor	0001
init	2	integer	Load control registers	0010
step	–	integer	Process one image frame for each enabled polarization and stop	0010
stepn	1	integer	Process n image frames for each enabled polarization and stop	0100
loadref	31,744	complex	Load reference kernels	0101
loadequal	8,192	complex	Load equalization weight	0110
loadiqeven	48	real	Load I/Q filter weights for even samples	0111
loadiqodd	48	real	Load I/Q filter weights for odd samples	1000
loadrcs	2048	real	Load RCS weights	1001

3. Test Bench

The VHDL test bench drives the processor and performs checks on the output.

The test bench reads processor commands from the `commands.dat` file and setup data from the files described in Section 5.5 and creates a stream of data for the processor C/D port described in BM1.

It checks that each setup file contains enough data and halts if it is insufficient. (The processor model does no checking of this data.)

The test bench supplies data to the processor from the `radar.asc` file and generates filler data of all zeros for a number of words specified by the FILLCOUNT variable described in Section 5.1.3. It writes into four different files for the different polarizations: `hhimage.asc`, `hvimage.asc`, `vhimage.asc`, `vvimage.asc`. Writing of output data to files can be disabled with the `output_image` generic described in Table 2. The processor still sends data to the processor when `output_image` is false.

The test bench compares the magnitude of each pixel from the processor against the magnitude of the same pixel in the appropriate comparison file, ie. `hhcomp.asc`, `hvcomp.asc`, `vhcomp.asc`, or `vvcomp.asc`. The comparison function is described in BM1. The error limit, in dB, is set in the test bench with the `thresh_db` generic described in Table 2. For each frame of output data the test bench writes to the `testbench.log` file the number of pixels which exceeded the specified limit and the location and error, in dB, of the pixel with largest error. There is no comparison made of header data. The `hdrcmp` program can be used for this purpose.

The latency of the processor is compared with a limit value set with the `latency` generic described in Table 2. The measured latency is recorded in the `testbench.log` file along with an indication if the value is outside a small window around the specified limit.

4. VHDL Simulation

The simulator has been written in VHDL in compliance with IEEE1076-1987 using the Vantage simulator.

4.1 Organization

Figure 2 depicts the organization of the high level VHDL models.

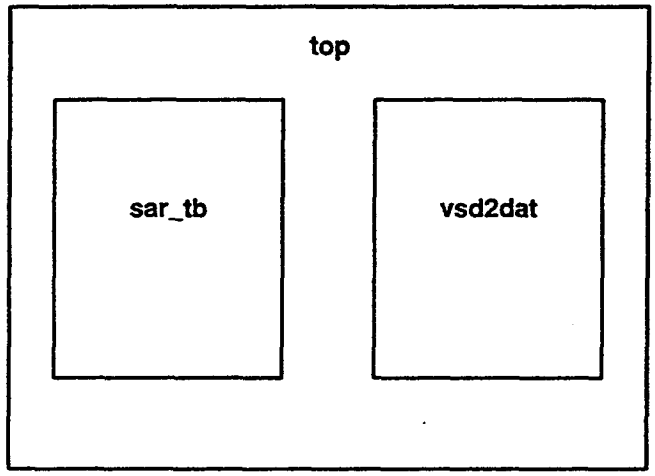


Figure 2. Top level of the VHDL simulation.

4.2 Libraries

The VHDL code uses the following libraries:

- `std.textio`
- `ieee.std_logic_1164`

and the following libraries are created:

- `math`
- `vsd2dat.lib`

4.3 VHDL Files

The VHDL files in this Executable Requirements are described in the following sections:

4.3.1 arrays.vhd

Contains package `arrays` comprising several array and other type definitions.

4.3.2 bvarithmetic.vhd

`bvarithmetic` contains a number of functions for conversion between bit vectors and integers. It is copyrighted by Synopsys and may be used and distributed without restriction provided that the copyright statement is not removed from the file and that any derivative work contains the copyright statement.

4.3.3 cdft.vhd

The `dsp` package has procedures for forward and inverse DFT of complex-value vectors.

4.3.4 complex.vhd

The `complex` package contains a type definition for a complex record and procedures for complex multiplication, complex-real multiplication and magnitude squared calculation.

4.3.5 frm_hdr.vhd

The `frm_hdr` package has three procedures for reading, writing and manipulating the data in the header of the radar data.

4.3.6 hex_textio.vhd

This file is derivative of a source file from Synopsys and so must contain the Copyright notice as described in Section 4.3.2. The `hex_textio` package contains the `hread` and `hwrite` procedures which are textio procedures that convert between hex and bit vector representations. The package also includes two procedures that convert between bit vector and real representations.

4.3.7 math_body.11.21.vhd, math_head.11.21.vhd

These files contain the `math_complex` package. It is derivative of the draft files from the IEEE VHDL Math Package Study Group which is NOT endorsed or approved by the IEEE. Major modifications were made to the `atan2` and `round` functions by Adam Rosenberg of Protocol, a Division of Zycad.

4.3.8 math_lib.vhdl

This file contains a package of generic math functions that were originally written in Ada and translated to VHDL by ETDL, U.S. Army, Fort Monmouth, NJ. Functions and procedures used from this package include `log`, and `sin` operators. This package is copyrighted by Donald F. Hanson and is provided for unrestricted use provided that the copyright notice is not removed.

4.3.9 mathmac.vhd

The `mathmac` package contains min and max functions for integers.

4.3.10 mempak.vhd

The `mempak` package contains many conversion functions. It was developed by Protocol.

4.3.11 read_adts.vhd

The `read_adts` package contains the `cmp_barker` and `read_pri` functions which are used in reading input data. `Cmp_barker` is overloaded to handle both a file and a data stream.

4.3.12 sar_tb.vhd

This file has the entity and behavioral architecture of the test bench. It performs the functions described in Chapter 3. The module generates two clocks, the 12.5 MHz optical fiber clock and the 4.56 MHz input data reference.

The interface of this module consists of several signals and generics. The signals are all of `std_logic` or `std_ulogic` type and are:

- clk - the 12.5 MHz fiber optic clock
- loadcmd - signal used to indicate presence of a command on the diagnostic (control) port
- datain - data received from the SAR processor (40 bits) defined in the BM1 document
- dataout - data sent to the SAR processor (40 bits) defined in the BM1 document
- diags - 32 bit diagnostic (control) signal used for initialization of the SAR processor
- datainstrobe - strobe for signal datain
- dataoutstrobe - strobe for signal dataout
- diagstrobe - strobe used for diagnostic (control) port data

The generics used in `sar_tb.vhd` are listed in Table 2 along with their function and value in `top.vhd` as delivered.

TABLE 2

Generics in sar_tb.vhd

Generic	Function	Value in top.vhd
fscr	Command file name	commands.dat
asckrn	Kernel file name	kernel.asc
frcs	RCS Data file name	rcs.asc
feq	EQ Data file name	equalize.asc
fiqo	Odd FIR Data file name	iqodd.asc
fiqe	Even FIR Data file name	iqeven.asc
fin	SAR Data file name	radar.asc
fouthh	Output Image Data file name – HH polarity	hhimage.asc
fouthv	Output Image Data file name – HV polarity	hvimage.asc
foutvh	Output Image Data file name – VH polarity	vhimage.asc
foutvv	Output Image Data file name – VV polarity	vvimage.asc
fp1	Comparison file name – HH polarization	hhcomp.asc
fp2	Comparison file name – HV polarization	hvcomp.asc
fp3	Comparison file name – VH polarization	vhcomp.asc
fp4	Comparison file name – VV polarization	vvcomp.asc
thresh_db	The error limit for output comparison	-103
refmax	Reference peak used in comparison function	1.4736e18
latency	Processor latency limit	1 sec
compare	Enables comparison function	TRUE
output_image	Enables writing of output to files	TRUE
flog	Output Log data file name	testbench.log

TABLE 3

Generics in vsd2dat.vhd

Generic	Function	Value in top.vhd
latency	Processor latency	1 sec

4.3.13 sarp.vhd

The sar package contains constant declarations.

4.3.14 top.vhd

This file is used to instance and configure the processor under test and the test bench. This is the highest level in the design hierarchy and the location of all generic parameters that are user definable. The generic parameters and signals contained in this design unit are defined in Sections 4.3.12 and 4.3.15.

4.3.15 vsd2dat.vhd

This file contains the entity and behavioral architecture for the SAR processor. It implements the functionality described in Chapter 2. Signals used by this design unit are defined in Section 4.3.12. The one generic is shown in Table 3.

5. File formats

Files are described using the default names, where appropriate, from the file name generics described in Section 4.3.12.

5.1 Input Data

5.1.1 radar.vsd

The Lincoln Laboratory Benchmark team starts with radar data in the format of this file. It is listed here only to explain certain terminology in this document.

5.1.2 radar.adts

This is radar data in a binary representation in the format described in Figure 5 of BM1 but with only bits 31-00 of the 40 shown there.

5.1.3 radar.tb

This is the data of radar.adts but with the VARIABLE FILLER WORDS replaced by one word which is a count of the filler words. The format is shown in Figure 3. Lincoln Laboratory will ensure that the data for each Pulse Repetition Interval will comprise data for four polarizations and that each polarization will have a FillCount word, 20 Barker words and 2032 data words. The data is in binary representation. The Executable Requirement includes radar data files in this format.

5.1.4 radar.asc

The VHDL test bench can read only ASCII files. Radar.asc is the data of radar.tb but in a hexadecimal representation of the binary data. The test bench can read files with several words on a line, we have used six. The user will use the program bin2ascii to convert radar.tb into radar.asc. A radar.asc file is twice as large as a radar.tb file.

5.2 Output Data

The test bench creates a separate file for each enabled polarization with consecutive frames in sequence in each file. The format of the data is described in Figure 11 of BM1 but only bits 31:00 are recorded and the files are hexadecimal representations of the binary data.

The four possible files are: hhimage.asc, hhimage.asc, hhimage.asc, and hhimage.asc.

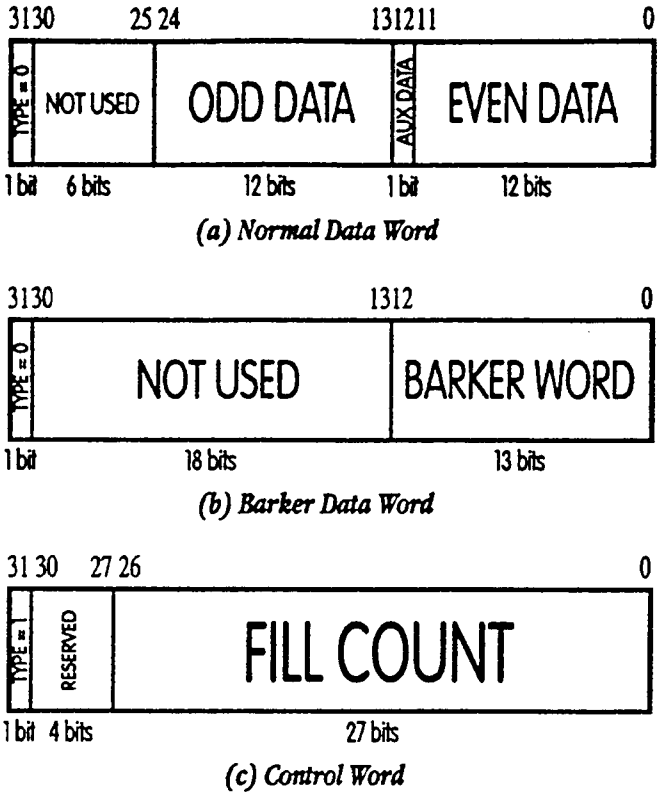


Figure 3. Format of data in radar.adts file.

5.3 Comparison Data

The test bench comparison function requires comparison files with format identical to the output data files described in Section 5.2. Comparison files will be delivered with the correct format in binary representation with `bin` file extension. The user must convert these files to hexadecimal representation of the binary with `bin2ascii`.

The four required files are: `hhcomp.asc`, `hvcomp.asc`, `vhcomp.asc`, and `vvcomp.asc`.

5.4 Command

Commands for the simulator are supplied in `commands.dat` which can contain the commands of Table 1. This file is in human readable ASCII and commands can be in either upper or lower case. Data lines for `init` and `stepn` follow the command. The test bench obtains data for the `setup` commands from the files described in Section 5.5. Example files are shown in Appendix A and Appendix B.

5.5 Setup

Setup files are supplied in the formats described in BM1, Section 2.2.1, in binary representation with `bin` file extension. The user must convert these files to hexadecimal representation of the binary with `bin2ascii` and give them the same name with an `asc` extension.

The files which are supplied are: `kernel.bin`, `equalize.bin`, `iqeven.bin`, `iqodd.bin`, and `rcs.bin`.

5.6 Log

The testbench creates a log file which includes an echo of the commands in `commands.dat`, the data entry and exit times which define latency and the output of the latency check and data comparison. It is called `testbench.log`.

6. Execution of Executable Requirement

6.1 Computer Requirements

6.2 Swap Space

Running Vantage Spreadsheet 5.011 in batch mode under SunOS 4.1.3, this simulation requires 168 mbytes of swap space without any buffering of output data. With the maximum latency of 3 seconds and with `stepN` of four or more, four complete frames of output data must be buffered for each enabled polarization. Data is buffered in `real` form so with the Vantage double precision `reals` each output frame requires:

$$2048 \times 512 \times 2 \times 8 + 78 \times 4 = 16,777,528 \text{ bytes.}$$

The maximum swap space for output buffering is twelve times this or about 201 MByte. Total maximum required swap space, then, is 369 Mbyte beyond whatever the system and other running programs require.

6.3 Physical Memory

Execution of the simulation was done on a Sun SPARC 10/51 with 64 MB of physical memory. Swapping activity as monitored with the Sun Perfmeter did not seem high, typically about 5 pages a second.

6.4 Disk Size

The size of the distribution is given in Chapter 9. Additional space is needed for compilation and for working files. In the HEX representation each frame of input data is 32 MByte and each frame of output data is about 16 MB.

6.5 Running Times

With Vantage SpreadSheet running on a Sun SPARC 10/51 with 64 MByte of physical memory the simulator requires 12.5 minutes to process one polarization of one frame of data. (A C double-precision program requires 0.56 minutes for the same processing.) The time to read and write data files and perform the necessary data conversions is quite long. The test bench converts the HEX encoded radar data to a 40-bit `std_logic` vector with three replications of the auxiliary bits and generates filler data between frames of data from a one-word specification in the file. Typical data has about one third as many fill as data words. The processor accepts bit vectors, detects the Barker code framing information, extracts and parallelizes the auxiliary data, extracts two 12-bit integers from each data word and converts them to `real` variables. All of this takes

about 84 minutes per frame with one polarization enabled. However, most of the operations are done on all the input data so enabling of more polarizations does not have a large impact.

At the output, the processor model converts `real` values to `std_logic`. The test bench HEX encodes the corresponding `bit_vector` and writes it to a file. For data comparison, the test bench reads the reference file, converts the two variables to `real`, and performs the comparison. Writing and comparison together require about 45 minutes per frame for each enabled polarization.

Some examination has been done of the execution times by recording running times with changes to the code which changed relative amounts of work being done. The following experiments were done to examine timing at the input of the simulation:

1. Changed `vsd2dat.vhd` so that `read_pri` is called only once every 32 pulses and for the other 31 pulses the data is reused. From this experiment we determined that the `read_pri` function takes 9.9 seconds per pulse (this includes the testbench operations, of course) and the algorithmic processing 0.4 seconds.
2. Created a `radar.tb` file which contained only enough data for the auxiliary polarization information and had the testbench create the remaining data. This reduced disk reads per pulse by 98% and did not have a dramatic effect on running time.
3. The data type on the data port between test bench and processor was changed from `std_logic_vector` to `bit_vector` and execution time, as measured by wall clock timing and the UNIX `time` command, increased.
4. In `sar_tb.vhd`, instead of using the FillCount variable, only two filler words were generated for each pulse. This decreased time per pulse.

The results of the first experiment make it clear that the focus on reducing running time in the input part of the simulation should be on economical movement of data, with the necessary format changes, from disk to processor model. For that reason, results of the other experiments are confusing and disappointing. Further experiments are planned and any observations or suggestions are solicited.

At the output we have observed that disabling both writing of output data to disk files and the comparison function does not have a dramatic effect on output time. Conversion of `real` data to `bit_vector` format is probably responsible for the long times.

7. Data Sets

7.1 M143

The m143 data set is unclassified data from the Advanced Detection Technology Sensor. It is supplied with four frames and all four polarizations in each frame. Setup data is supplied. The output image, as displayed with `dat2display`, is shown in Figure 4. This image was created with the C-single precision program from HH data; images from the C-double precision program and the VHDL simulator look the same, as do the other polarizations.

7.2 Test

The test data set is synthesized data which produces an image of seven reflectors as shown in Figure 5. This data is to be processed with the setup data supplied for m143. It is supplied with four frames of data and four polarizations but the polarizations are identical.

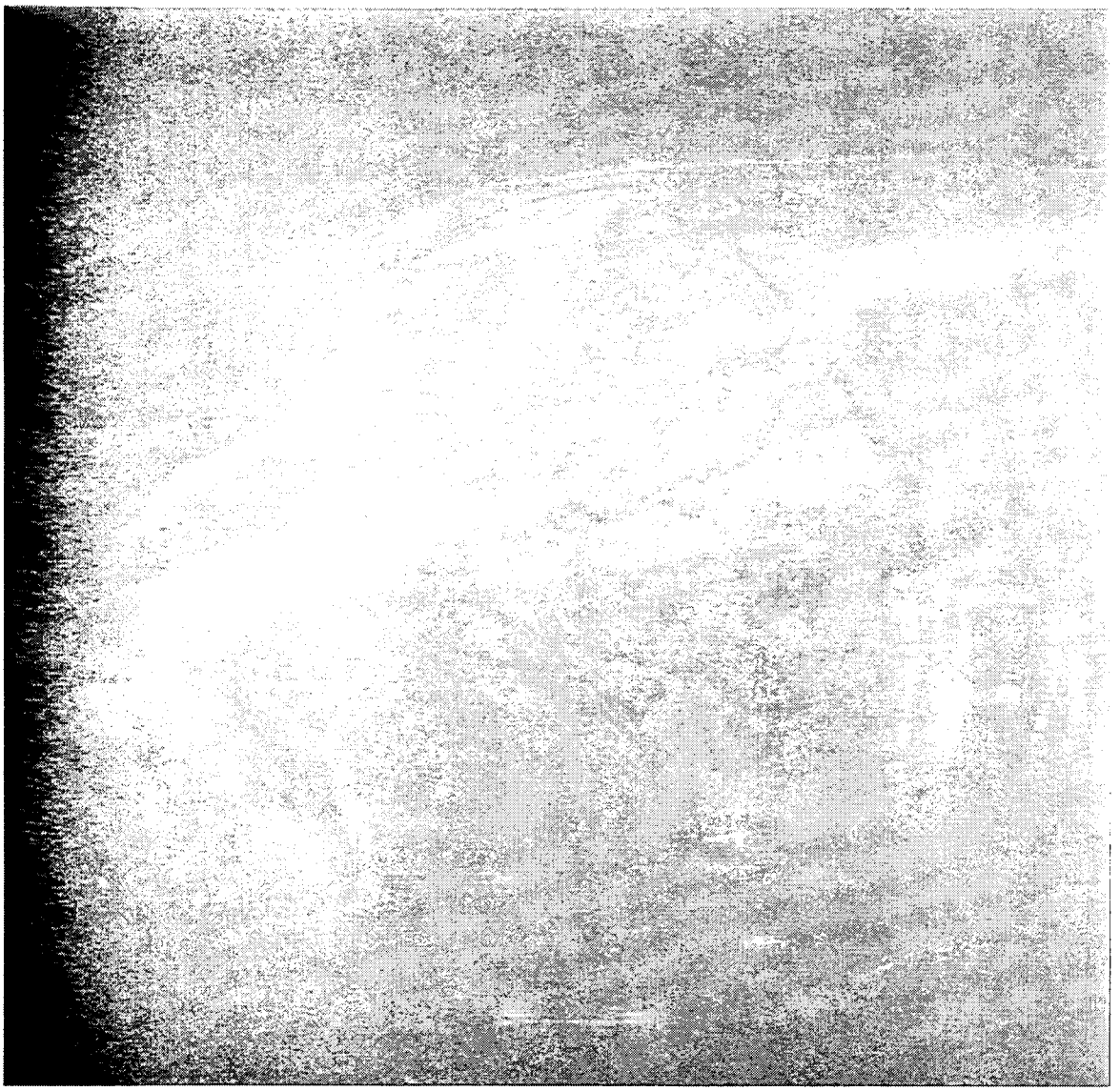


Figure 4. Image from m143 radar data created by a C-language single precision program and displayed with `dat2display` *using mindb=100 and range=70 parameters.*

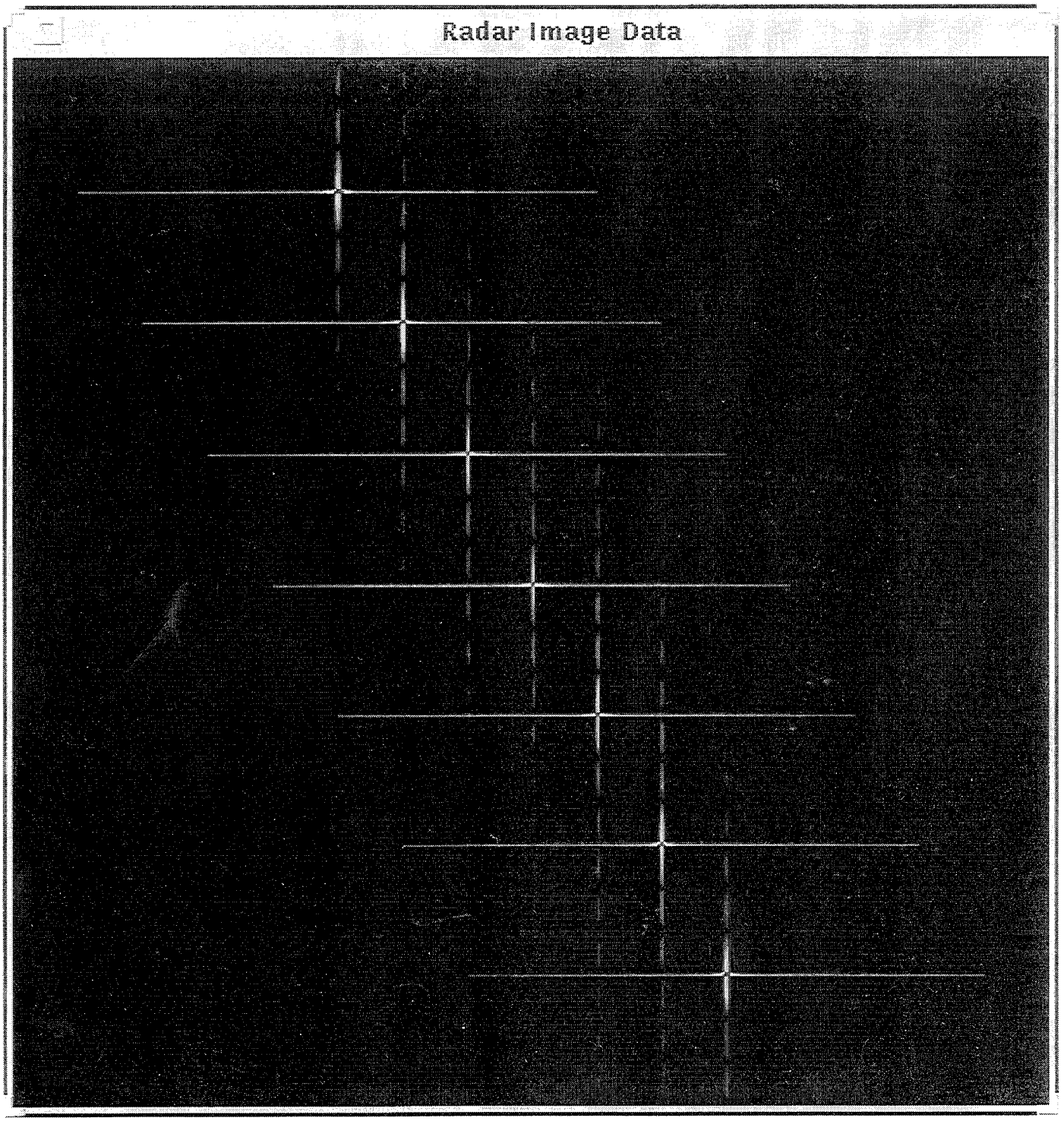


Figure 5. Image from synthesized test data created by a C-language single precision program and displayed with `dat2display` *using mindb=90 and range=70 parameters.*

8. Utility Programs

Several C programs are supplied for data conversion, comparison and display. They are described here.

8.1 Data Conversion

bin2ascii — Converts any 32-bit binary data structure (`int`, `float`) to its 32-bit hexadecimal ASCII representation.

ascii2bin — Converts 32-bit hexadecimal ASCII representations to 32-bit binary data structures.

8.2 Comparison

histcmp — This application reads two test bench output files in the **.bin** format and computes the log-magnitude error between the two images on a pixel by pixel basis according to the accuracy criterion established in the Benchmark 1 Technical Description. A histogram of the percentage of failed pixels as a function of error threshold is then generated and sent to *stdout*.

hdrcmp — This program compares the frame headers of two files in the **.bin** format outputted by the SAR processor and prints the number of differing header words. A verbose option is also available so that header information for both files may be examined.

8.3 Output Print

histcmp-pr — This application is a modification of histcmp which enables output of the value of selected pixels in both image files. If neither of the index options are used this program behaves identically to histcmp and prints no values. Both the selected pixel values and the histogram are sent to *stdout*.

histcmp-err — This application is a modification of histcmp which outputs the complex values of the input and reference data and error value for the four pixels with largest error. It does not output a histogram. Output is to *stdout*.

8.4 Display

dat2display — A simple viewer using the OPEN LOOK interface in an X11R5 environment, this application reads processor output in the .bin format and displays it as a function of pixel intensity according to user-defined color mapping.

8.5 Compilation

The distribution includes two subprograms `read_adts.c` and `frm_hdr.c` and the header file `sar.h`. There is a Makefile. For `dat2display`, path names to some X resources must be set in the Makefile.

8.6 Usage Chart

This chart describes the command-line options for the utility programs.

Option	Type	Default	Description
ascii2bin			
`-i`	*char[]*	none	Specify the input ASCII file.
`-o`	*char[]*	none	Specify the output binary file.
bin2ascii			
`-i`	*char[]*	none	Specify the input binary file.
`-o`	*char[]*	none	Specify the output ASCII file.
`-wpl`	*int*	6	Number of words per output ASCII scan line.
dat2display			
`-i`	*char[]*	none	Specify the input **.bin** file.
`-nframe`	*int*	4	Number of frames to display.
`-nskip`	*int*	0	Number of initial frames to skip in input file.
`-mindb`	*float*	60.0	Minimum pixel value in dB.
`-range`	*float*	100.0	Range in dB of pixel values above minimum.
hdrcmp			
`-i`	*char[]*	none	Specify the input **.bin** file.
`-ref`	*char[]*	none	Specify the reference **.bin** file.
`-iframe`	*int*	1	Input frame to compare.
`-rframe`	*int*	1	Reference frame to compare.
`-verb`		terse	Print verbose information about headers.
histcmp			
`-i`	*char[]*	none	Specify the input **.bin** file.
`-ref`	*char[]*	none	Specify the reference **.bin** file.
`-iframe`	*int*	1	Input frame to compare.
`-rframe`	*int*	1	Reference frame to compare.

Option	Type	Default	Description
histcmp-pr			
`-i`	*char[]*	none	Specify the input **.bin** file.
`-ref`	*char[]*	none	Specify the reference **.bin** file.
`-iframe`	*int*	1	Input frame to compare.
`-rframe`	*int*	1	Reference frame to compare.
`-rindex`	*int*	0	Range index of pixel to print.
`-aindex`	*int*	0	Azimuth index of pixel to print.
`-neigh`	*int*	0	Extent of neighbor pixels to print.
histcmp-err			
`-i`	*char[]*	none	Specify the input **.bin** file.
`-ref`	*char[]*	none	Specify the reference **.bin** file.
`-iframe`	*int*	1	Input frame to compare.
`-rframe`	*int*	1	Reference frame to compare.

9. Executable Requirements Package

The UNIX tar file has this directory organization:

```
bm1-exreq
 |
 +- data
 |   |
 |   +- m143
 |   |   |
 |   |   +- C-double
 |   |   +- C-single
 |   |
 |   +- test
 |       |
 |       +- C-double
 |       +- C-single
 |
 +- doc
 +- runs
 |   |
 |   +- mentor
 |   +- vantage
 |
 +- simulator
 +- utility
     |
     +- bin
     +- src
```

When the tape is untarred the `bm1-exreq` directory is approximately 460 MBytes in size.

Following is the output of doing `ls -lR` on one version of the tar file. This list should not be considered definitive, a listing will be delivered with the tape.

```
total 6
-rw-r--r--  1 anderson      869 Aug  5 15:57 README
drwxr-xr-x  4 anderson      512 Jul 22 16:34 data/
drwxr-xr-x  2 anderson      512 Aug  5 16:49 doc/
```

```
drwxr-xr-x  4 anderson      512 Aug  5 15:57 runs/
drwxr-xr-x  2 anderson      512 Aug  5 16:00 simulator/
drwxr-xr-x  4 anderson      512 Jul 22 15:46 utility/

bm1-exreq/data:
total 2
drwxr-xr-x  4 anderson      512 Jul 22 16:24 m143/
drwxr-xr-x  4 anderson      512 Aug  5 15:11 test/

bm1-exreq/data/m143:
total 66078
drwxr-xr-x  2 anderson      512 Jul 22 16:31 C-double/
drwxr-xr-x  2 anderson      512 Jul 22 16:28 C-single/
-rw-r--r--  1 anderson      342 Jul 22 16:31 README
-rw-r--r--  1 anderson      220 Jul 22 16:20 convertb2a.scr
-rw-r--r--  2 anderson    65536 Jul 15 11:18 equalize.bin
-rw-r--r--  2 anderson      224 Jul 22 15:17 iqeven.bin
-rw-r--r--  2 anderson      224 Jul 22 15:16 iqodd.bin
-rw-r--r--  2 anderson   253952 Jul 15 11:16 kernel.bin
-rw-r--r--  2 anderson 67272704 Jul 15 11:11 radar.tb
-rw-r--r--  2 anderson     8192 Jul 15 11:17 rcs.bin

bm1-exreq/data/m143/C-double:
total 131202
-rw-r--r--  1 anderson      336 Jul 22 16:32 README
-rw-r--r--  1 anderson      160 Jul 22 15:50 convertb2a.scr
-rw-r--r--  1 anderson 33555712 Jul 22 15:29 hhimage.bin
-rw-r--r--  1 anderson 33555712 Jul 22 15:33 hvimage.bin
-rw-r--r--  1 anderson 33555712 Jul 22 15:36 vhimage.bin
-rw-r--r--  1 anderson 33555712 Jul 22 15:39 vvimage.bin

bm1-exreq/data/m143/C-single:
total 131202
-rw-r--r--  1 anderson      337 Jul 22 16:32 README
-rw-r--r--  1 anderson      160 Jul 22 15:50 convertb2a.scr
-rw-r--r--  2 anderson 33555712 Jul 20 10:14 hhimage.bin
-rw-r--r--  2 anderson 33555712 Jul 20 10:17 hvimage.bin
-rw-r--r--  2 anderson 33555712 Jul 20 10:21 vhimage.bin
-rw-r--r--  2 anderson 33555712 Jul 20 10:26 vvimage.bin

bm1-exreq/data/test:
```

```
total 65747
drwxr-xr-x  2 anderson      512 Aug  5 15:14 C-double/
drwxr-xr-x  2 anderson      512 Aug  5 15:13 C-single/
-rw-r--r--  1 anderson      160 Aug  5 15:12 README
-rw-r--r--  2 anderson 67272704 Jul 29 08:23 test.tb

bm1-exreq/data/test/C-double:
total 32801
-rw-r--r--  1 anderson      145 Aug  5 15:14 README
-rw-r--r--  2 anderson 33555680 Jul 28 16:37 hhimage.bin

bm1-exreq/data/test/C-single:
total 32801
-rw-r--r--  1 anderson      145 Aug  5 15:14 README
-rw-r--r--  2 anderson 33555680 Jul 29 08:30 hhimage.bin

bm1-exreq/doc:
total 1464
-rw-r--r--  1 anderson  1487074 Aug  5 16:48 users_manual.ps

bm1-exreq/runs:
total 2
drwxr-xr-x  2 anderson      512 Aug  5 16:46 mentor/
drwxr-xr-x  2 anderson      512 Aug  5 16:45 vantage/

bm1-exreq/runs/mentor:
total 27
-rw-r--r--  1 anderson       56 Aug  5 15:59 README
-rw-rw-rw-  2 anderson       72 Aug  4 16:22 commands.dat
-rw-rw-rw-  2 anderson      553 Aug  4 16:24 hh-s.cmp
-rw-rw-rw-  2 anderson      551 Aug  4 16:24 hv-s.cmp
-rw-rw-rw-  2 anderson     4613 Aug  4 16:58 run.scr
-rw-rw-rw-  2 anderson     1874 Aug  4 16:24 run.scr.log
-rw-r--r--  2 anderson      820 Aug  4 16:59 sar.make
-rw-r--r--  2 anderson    13127 Aug  4 16:59 sar.make.log
-rw-rw-rw-  2 anderson      911 Aug  4 16:24 testbench.log
-rw-rw-rw-  2 anderson      553 Aug  4 16:24 vv-s.cmp

bm1-exreq/runs/vantage:
total 49
-rw-r--r--  1 anderson       58 Aug  5 16:46 README
```

```
-rw-r--r--  2 anderson      72 Jul 22 05:41 commands.dat
-rw-r--r--  2 anderson    1648 Jul 22 14:51 hh1-single.cmp
-rw-r--r--  2 anderson    1729 Jul 22 14:53 hh2-single.cmp
-rw-r--r--  2 anderson    9797 Jul 22 05:30 mon-21jul.log
-rw-r--r--  2 anderson    8471 Jul 22 14:41 perfmeter.log13418
-rwxr-xr-x  1 anderson     867 Jul 22 15:33 run.scr*
-rw-r--r--  2 anderson   12842 Jul 22 03:34 run.scr.log
-rwxr-x---  2 anderson    1156 Jul 15 15:15 sar.make*
-rw-r--r--  1 anderson    7262 Jul 21 17:37 sar.make.log

bm1-exreq/simulator:
total 309
-rw-r-----  2 anderson    1740 Jul 21 17:32 arrays.vhd
-rw-r-----  2 anderson   11733 Jul 15 09:47 bvarithmetic.vhd
-rw-r-----  2 anderson    2537 Jul 21 17:31 cdft.vhd
-rw-r-----  2 anderson    1202 Jul 21 17:29 complex.vhd
-rw-r-----  2 anderson    8806 Jul 21 17:26 frm_hdr.vhd
-rw-r-----  2 anderson    6978 Jul 15 09:47 hex_textio.vhd
-rw-r-----  2 anderson   41162 Jul 15 09:47 math_body.11.21.vhd
-rw-r-----  2 anderson    8008 Jul 15 09:47 math_head.11.21.vhd
-r--r-----  2 anderson  125371 Jul 15 09:48 math_lib.vhdl
-rw-r-----  2 anderson     842 Jul 21 16:57 mathmac.vhd
-rw-r-----  2 anderson   11261 Jul 21 17:17 mempak.vhd
-rw-r-----  2 anderson   21082 Jul 21 17:14 read_adts.vhd
-rwxr-x---  2 anderson     229 Jul 18 16:48 sar_tb.make*
-rw-r-----  2 anderson   31784 Jul 21 16:44 sar_tb.vhd
-rw-r-----  2 anderson     756 Jul 21 17:17 sarp.vhd
-rw-r-----  2 anderson    3936 Jul 28 15:58 top.vhd
-rw-r--r--  2 anderson   18082 Jul 21 16:37 vsd2dat.vhd

bm1-exreq/utility:
total 3
-rw-r--r--  1 anderson     259 Aug  5 15:16 README
drwxr-xr-x  2 anderson     512 Jul 22 15:43 bin/
drwxr-xr-x  2 anderson     512 Aug  5 14:50 src/

bm1-exreq/utility/bin:
total 0

bm1-exreq/utility/src:
total 78
```

```
-rw-r--r--  2 anderson     1276 Aug  5 14:48 Makefile
-rw-r--r--  2 anderson     2283 Jul 15 10:33 ascii2bin.c
-rw-r--r--  2 anderson     2304 Jul 15 10:33 bin2ascii.c
-rw-r--r--  2 anderson     5872 Jul 15 10:33 dat2display.c
-rw-r--r--  2 anderson    23964 Jul 15 10:54 display_x.c
-rw-r--r--  2 anderson     3618 Jul 15 10:41 frm_hdr.c
-rw-r--r--  2 anderson     4469 Jul 15 10:33 hdrcmp.c
-rw-r--r--  2 anderson     5513 Aug  5 14:47 histcmp-err.c
-rw-rw-rw-  2 anderson     6331 Aug  2 08:30 histcmp-pr.c
-rw-r--r--  2 anderson     5286 Jul 15 10:33 histcmp.c
-rw-r--r--  2 anderson     9035 Jul 15 10:41 read_adts.c
-rw-r--r--  2 anderson     2593 Jul 15 10:43 sar.h
```

10. Notes

Gathered here are some notes on the Executable Requirement.

1. Squint angle— The procedure `read_frm_hdr` in `frm_hdr.vhd` computes squint angle from auxiliary information. This functionality is not required by BM1 and the information is not used in this model.

2. Comparison files— The testbench requires that all four comparison files be present even if not all polarizations are enabled. If a polarization is not enabled then the file for that polarization could be a copy (or link) to some other comparison file.

3. Simulation time step— The VHDL processes for the 4.56, 8.0 and 12.5 MHz clocks use picosecond time steps to derive precise clock periods. If a simulator counts simulation time with a 32 bit integer then the simulation will not continue beyond the first few input pulses. It is certainly permissible to change the clock period times to nanoseconds and have slightly different clock periods. However, even with this change, Mentor QuickVHDL version 8.2 seems to have a limit of $2^{31} - 1$ time steps which is not enough for two frames with three enabled polarizations. Mentor QuickVHDL version 8.4 apparently uses a 64 bit integer for counting time steps so there will be no problem with using ps in the clock processes.

4. Fills disk— If the test bench cannot find a header in a comparison file, for instance, if the file is misformatted or too short, it will put out error messages continuously and may fill up the disk. This is a bug.

APPENDIX A
Example Run With Vantage

A.1 Environment

This example run with the m143 data set was done with Vantage Batch Simulator v5.011 on a Sun SPARC 10/51 with 64 MByte of physical memory in SunOS 4.1.3. There were no other users on the system to the best of my knowledge. Relevant files are in the distribution directory `bm1-exreq/runs/vantage`.

A.2 Compile

The `sar.make` file shown in Figure A-1 was used to compile the simulator.

```
vanlibcreate math math
vanlibcreate vsd2dat.lib vsd2dat
analyze -src math_head.11.21.vhd -lib math -nobndchk -dbg 0
analyze -src math_body.11.21.vhd -lib math -nobndchk -dbg 0
analyze -src math_lib.vhdl -lib math -nobndchk -dbg 0
analyze -src mathmac.vhd -lib vsd2dat.lib -nobndchk -dbg 0
analyze -src complex.vhd -lib vsd2dat.lib -libieee -nobndchk -dbg 0
analyze -src arrays.vhd -lib vsd2dat.lib -libieee -nobndchk -dbg 0
analyze -src mempak.vhd -lib vsd2dat.lib -libieee -nobndchk -dbg 0
analyze -src bvarithmetic.vhd -lib math -nobndchk -dbg 0
analyze -src hex_textio.vhd -lib vsd2dat.lib -lib math -libieee -nobndchk -dbg 0
analyze -src cdft.vhd -lib vsd2dat.lib -lib math -libieee -nobndchk -dbg 0
analyze -src sarp.vhd -lib vsd2dat.lib -libieee -nobndchk -dbg 0
analyze -src read_adts.vhd -lib vsd2dat.lib -libieee -lib math -nobndchk -dbg 0
analyze -src frm_hdr.vhd -lib vsd2dat.lib -libieee -lib math -nobndchk -dbg 0
analyze -src sar_tb.vhd -lib vsd2dat.lib -libieee -lib math -nobndchk -dbg 0
analyze -src vsd2dat.vhd -lib vsd2dat.lib -libieee -lib math -nobndchk -dbg 0
analyze -src top.vhd -lib vsd2dat.lib -libieee -lib math -nobndchk -dbg 0
```

Figure A-1. `sar.make` *for example Vantage run.*

A.3 Run Parameters

The commands.dat file is shown in Figure A-2. Polarizations HH, HV and VV are enabled and two frames of data are to be processed. The thresh_db generic in top.vhd was set to -170 dB to ensure detection of some errors.

```
reboot
InIt
11
8
loadrcs
loadequal
loadiqeven
loadiqodd
loadref
stepN
2
```

Figure A-2. commands.dat *for example Vantage run.*

The script file, called run.scr, shown in Figure A-3 was used to set up files and do the simulation. I did csh run.scr >& run.log to save a record of the run. (I should have also echoed all the script commands for a complete record.) The Vantage Batch Simulator also creates a file with simulator messages. (The directory name for the comparison files is different from what is on the distribution tape.) The two egrep commands are used to make a record of generics which are often changed and the analyze -src top.vhd .. is included to ensure that the current generic values are loaded into the simulator. The UNIX csh time command is used to have a record of running time.

A.4 Results

The testbench.log file created by this run is shown in Figure A-4.

An edited version of the output of csh run.scr is shown in Figures A-5 and A-6. The full file is in the runs/appa directory. The running time was 9:47 hours and the output of time at the end of the log file shows that the simulator averaged 96% utilization of the computer. The outputs

```
rm hhcomp.asc
ln -s ../data/m143/images-single/hhimage.asc hhcomp.asc
rm hvcomp.asc
ln -s ../data/m143/images-single/hvimage.asc hvcomp.asc
rm vhcomp.asc
ln -s ../data/m143/images-single/vhimage.asc vhcomp.asc
rm vvcomp.asc
ln -s ../data/m143/images-single/vvimage.asc vvcomp.asc
rm iqeven.asc
ln -s ../data/m143/iqeven.asc iqeven.asc
rm iqodd.asc
ln -s ../data/m143/iqodd.asc iqodd.asc
rm kernel.asc
ln -s ../data/m143/kernel.asc kernel.asc
rm radar.asc
ln -s ../data/m143/radar.asc  radar.asc
rm rcs.asc
ln -s ../data/m143/rcs.asc  rcs.asc
rm equalize.asc
ln -s ../data/m143/equalize.asc  equalize.asc
egrep -e '^compon|latency' top.vhd
egrep 'compare' top.vhd
egrep 'output_image' top.vhd
analyze -src top.vhd -lib vsd2dat.lib -libieee -lib math -nobndchk -dbg 0
date
time vbsim -cfg "top_cfg" -lib vsd2dat.lib -libieee -lib math -until "complete" -sev "error"
date
```

Figure A-3. `run.scr` *for example Vantage run.*

```
reboot
init
11
8
loadrcs
loadequal
loadiqeven
loadiqodd
loadref
stepn
2
HH 512th pulse inputted to SAR at time =1106689640 NS
HH 512th pulse inputted to SAR at time =2215666200 NS
Data for HH polarization frame 1 was received at time =3106672700 NS
actual latency observed is =1999983060 NS
Performing HH comparison with reference data at time =3107132400 NS
Threshold was: -170
Number of pixels with error above threshold: 4491
Largest error at: range: 466 azimuth: 434 error: -158
Performing HV comparison with reference data at time =3369286160 NS
Threshold was: -170
Number of pixels with error above threshold: 5106
Largest error at: range: 0 azimuth: 241 error: -160
Performing VV comparison with reference data at time =3631439920 NS
Threshold was: -170
Number of pixels with error above threshold: 4917
Largest error at: range: 493 azimuth: 443 error: -157
Data for HH polarization frame 2 was received at time =4215649260 NS
actual latency observed is =1999983060 NS
Performing HH comparison with reference data at time =4216109040 NS
Threshold was: -170
Number of pixels with error above threshold: 21530
Largest error at: range: 1001 azimuth: 19 error: -157
Performing HV comparison with reference data at time =4478262800 NS
Threshold was: -170
Number of pixels with error above threshold: 21348
Largest error at: range: 1010 azimuth: 20 error: -156
Performing VV comparison with reference data at time =4740416560 NS
Threshold was: -170
Number of pixels with error above threshold: 22151
Largest error at: range: 1000 azimuth: 86 error: -157
```

Figure A-4. `testbench.log` *from example Vantage run.*

of a performance meter and the program `pstat -s`, which measures swap space usage, which were running at a 5 minute sampling interval, are in the `runs/appa` directory. The directory also has the outputs of `histcmp-pr` for this run which confirm the imaged errors reported in `testbench.log`.

```
component vsd2dat
                latency:time:=2000 ms);
component sar_tb
latency:time:=2000 ms;            --delay from 512th HH to output from SAR
compare:boolean:=TRUE; --/perform comparison against reference data?
output_image:boolean:=TRUE --/Write image data to external file?


    VHDL Compiler, Release 5.011
    Copyright (c) 1993, Vantage Analysis Systems, Inc.
    Working library VSD2DAT "vsd2dat.lib".
    --
    Compiling "top.vhd" line 1...
    Compiled entity VSD2DAT.TOP
    --
    Compiling "top.vhd" line 37...
    Compiled architecture VSD2DAT.TOP(A)
    --
    Compiling "top.vhd" line 105...
    Compiled configuration VSD2DAT.TOP_CFG of TOP(A)
    --
    3/3 design unit(s) compiled successfully.
    Syntax summary: 0 error(s), 0 warning(s) found.
Thu Jul 21 17:47:14 EDT 1994

Vantage Batch Simulator,  v5.011.  Release 5.011
Copyright (c) 1987-1993, Vantage Analysis Systems, Inc.

**Assertion NOTE:  A reboot has been requested
**Assertion NOTE:  Initializing polarization and # of FIR taps
**Assertion NOTE:  Loading RCS Weighting Coefficients
**Assertion NOTE:  Loading Equalization and Taylor Weighting Coefficients
**Assertion NOTE:  Loading even FIR coefficients
**Assertion NOTE:  Loading odd FIR coefficients
**Assertion NOTE:  Loading convolution kernels
**                  at delta 3 of time 37107 ns.
**Assertion NOTE:  SAR Processor is now running and waiting to accept data
**                  at delta 2 of time 164090 ns.
**Assertion NOTE:  Opening the input data
**Assertion NOTE:  Reading frame 1.
**Assertion NOTE:  Range processing.
**                  at delta 5 of time 164092 ns.
**Assertion NOTE:  512th pulse encountered
**                  at delta 4 of time 1107122720 ns.
```

Figure A-5. An abbreviated version of output of `run.scr` *from example Vantage run-part1.*

```
**Assertion NOTE:  Azimuth processing.
**                  at delta 4 of time 1108474 us.
**Assertion NOTE:  Reading frame 2.
**                  at delta 12583384 of time 1108474 us.
**Assertion NOTE:  Range processing.
**Assertion NOTE:  512th pulse encountered
**                  at delta 4 of time 2216099280 ns.
**Assertion NOTE:  Azimuth processing.
**                  at delta 4 of time 2217450640 ns.
**Assertion NOTE:  SAR Processing Completed, Waiting for Latency for data
                        output completion
**                  at delta 12583384 of time 2217450640 ns.
**Assertion NOTE:  HH found
**Assertion NOTE:  Performing HH comparison with reference data
**                  at delta 4 of time 3107132400 ns.
**Assertion NOTE:  Threshold was: -170
**Assertion NOTE:  Number of pixels with error above threshold: 4491
**Assertion NOTE:  Largest error at: range: 466  azimuth: 434 error: -158
**Assertion NOTE:  HV found
**Assertion NOTE:  Performing HV comparison with reference data
**Assertion NOTE:  Number of pixels with error above threshold: 5106
**Assertion NOTE:  Largest error at: range: 0  azimuth: 241 error: -160
**Assertion NOTE:  VV found
**Assertion NOTE:  Performing VV comparison with reference data
**Assertion NOTE:  Number of pixels with error above threshold: 4917
**Assertion NOTE:  Largest error at: range: 493  azimuth: 443 error: -157
**Assertion NOTE:  HH found
**Assertion NOTE:  Performing HH comparison with reference data
**Assertion NOTE:  Number of pixels with error above threshold: 21530
**Assertion NOTE:  Largest error at: range: 1001  azimuth: 19 error: -157
**Assertion NOTE:  HV found
**Assertion NOTE:  Performing HV comparison with reference data
**Assertion NOTE:  Number of pixels with error above threshold: 21348
**Assertion NOTE:  Largest error at: range: 1010  azimuth: 20 error: -156
**Assertion NOTE:  VV found
**Assertion NOTE:  Performing VV comparison with reference data
**Assertion NOTE:  Number of pixels with error above threshold: 22151
**Assertion NOTE:  Largest error at: range: 1000  azimuth: 86 error: -157
**Assertion FAILURE:  *****Simulation has completed successfully*****
**                  at delta 5 of time 5002560560 ns.
**End of simulation.
33988.8u 151.2s 9:47:09 96% 0+-476k 3583+3382io 45822pf+0w
Fri Jul 22 03:34:25 EDT 1994
```

Figure A-6. An abbreviated version of output of run.scr *from example Vantage run-part2.*

APPENDIX B
Example Run with Mentor QuickVHDL

B.1 Environment

This example run with the m143 data set was done with Mentor QuickVHDL version 8.2 on a HP 9000/735 with 208 MByte of physical memory, 1.2 GByte of swap space and the HP-UX operating system. There were no other users on the system although the working disk was connected by ethernet. Relevant files are in the distribution directory `bm1-exreq/runs/mentor`.

We have less experience with Mentor than with Vantage so some of this is rather tentative. Mentor QuickVHDL 8.2 uses 32 bit integers for counting time steps, and apparently, a signed integer since the largest time step is $2^{31} - 1$. The three clock processes were changed to define the clock period in nanoseconds with the result that the maximum time in one simulation run is 2.14 seconds. Reading in one frame (512 pulses) of data requires about 1 second and writing out one polarization of output data about 0.25 second. If latency is set at 0.1 seconds then processing one frame with three polarizations enabled takes about 1.85 seconds, which can be done with the present limit. I have been told that Mentor QuickVHDL version 8.4 uses 64 bit integers for counting time steps.

Mentor QuickVHDL 8.2 uses single precision floating point variables to represent VHDL `real` values. All indications are that the SAR algorithm is computing images correctly in the Mentor simulator. HOWEVER, in the present distribution the `realtobv` function found in `hex_textio.vhd` does not do correct conversion for real values close to a power of two when simulated in this QuickVHDL. This function is used at the output of the processor model to convert image data to bitvector form. One run with the debugger showed that the pixel in HH with largest error, according to the testbench, had a correct `real` value in the processor. Also, the calculation of threshold in `sar_tb.vhd` is not correct and is always creating a value of zero (or very close to it). We are investigating these problems and will forward a fix when it is available. I understand that Mentor 8.4 QuickVHDL represents VHDL `real` values with double precision. We will test the current distribution on Mentor 8.4 when it is installed in a short time.

B.2 Compile

The `sar.make` file shown in Figure B-1 was used to compile the simulator. I tried using a `-nodebug` option but the compiler did not accept it. Figure B-2 shows extracts from the compiler output, the full file is in `runs/mentor`. Despite the error messages the compilation seemed to be successful.

B.3 Run Parameters

The `commands.dat` file is shown in Figure B-3. Polarizations HH, HV and HV are enabled and one frame of data is to be processed. The `thresh_db` generic in `top.vhd` was set to -103 dB.

```
qvlib vsd2dat.lib
qvlib math
qvmap math ./math
qvmap vsd2dat ./vsd2dat.lib
qvcom -work math  -nocheck math_head.11.21.vhd
qvcom -work math -nocheck math_body.11.21.vhd
qvcom -work math -nocheck math_lib.vhdl
qvcom -work vsd2dat.lib -nocheck mathmac.vhd
qvcom -work vsd2dat.lib -nocheck complex.vhd
qvcom -work vsd2dat.lib -nocheck arrays.vhd
qvcom -work vsd2dat.lib -nocheck mempak.vhd
qvcom -work math -nocheck bvarithmetic.vhd
qvcom -work vsd2dat.lib -lib math -nocheck  hex_textio.vhd
qvcom -work vsd2dat.lib -lib math -nocheck cdft.vhd
qvcom -work vsd2dat.lib -libieee -nocheck sarp.vhd
qvcom -work vsd2dat.lib -nocheck read_adts.vhd
qvcom -work vsd2dat.lib -nocheck frm_hdr.vhd
qvcom -work vsd2dat.lib -nocheck sar_tb.vhd
qvcom -work vsd2dat.lib -nocheck vsd2dat.vhd
qvcom -work vsd2dat.lib -nocheck top.vhd
```

Figure B-1. `sar.make` *for example Mentor run.*

```
//  QuickVHDL qvcom v8.2_1.4.1b  Fri Aug 20 14:23:25 1993
// Error: The following file does not exist in the source directory:
     hex_textio.vhd. (from: Analysis/System-1076_semantic 0604)
// Error: The following file does not exist in the source directory:
     cdft.vhd. (from: Analysis/System-1076_semantic 0604)
// Error: The following file does not exist in the source directory:
     sarp.vhd. (from: Analysis/System-1076_semantic 0604)
WARNING[2]: sar_tb.vhd(842): Possible infinite loop: process has no wait statement.
```

Figure B-2. Extracts from `sar.make.log` *for example Mentor run.*

```
reboot
InIt
11
8
loadrcs
loadequal
loadiqeven
loadiqodd
loadref
stepN
1
```

Figure B-3. `commands.dat` *for example Mentor run.*

The script file called `run.scr`, shown in Figure B-4, was used to set up files and do the simulation. I did not know how to run `qvsim` as a command line job and did not save any useful run information in a file. The simulation run time was set near the maximum value in the simulator window. The two egrep commands are used to make a record of generics which are often changed and the `qvcom .. top.vhd` is included to ensure that the current generic values are loaded into the simulator. Run time was 3:52 hours.

B.4 Results

The `testbench.log` file created by this run is shown in Figure B-5. Note the large number of reported errors due to the problem with calculating threshold and the large error due to the `realtobv` problem.

Figure B-6 shows the output of `histcmp-err` for polarization HV where the comparison is with the output of a single-precision C program. The four largest errors are all at pixels where the output value is off by a factor of two.

```
rm hhcomp.asc
ln -s ../data/m143/C-single/hhimage.asc hhcomp.asc
rm hvcomp.asc
ln -s ../data/m143/C-single/hvimage.asc hvcomp.asc
rm vhcomp.asc
ln -s ../data/m143/C-single/vhimage.asc vhcomp.asc
rm vvcomp.asc
ln -s ../data/m143/C-single/vvimage.asc vvcomp.asc
rm iqeven.asc
ln -s ../data/m143/iqeven.asc iqeven.asc
rm iqodd.asc
ln -s ../data/m143/iqodd.asc iqodd.asc
rm kernel.asc
ln -s ../data/m143/kernel.asc kernel.asc
rm radar.asc
ln -s ../data/m143/radar.asc  radar.asc
rm rcs.asc
ln -s ../data/m143/rcs.asc  rcs.asc
rm equalize.asc
ln -s ../data/m143/equalize.asc  equalize.asc
egrep -e '^compon|latency' top.vhd
egrep 'compare' top.vhd
egrep 'output_image' top.vhd
qvcom -work vsd2dat.lib -nocheck top.vhd
date
time qvsim -t ns -lib vsd2dat.lib top_cfg
date
```

Figure B-4. `run.scr` *for example Mentor run.*

```
reboot
init
11
8
loadrcs
loadequal
loadiqeven
loadiqodd
loadref
stepn
1
HH 512th pulse inputted to SAR at time =1110231640 ns
Data for HH polarization frame 1 was received at time =1210216060 ns
actual latency observed is =99984420 ns
Minimum latency time violation has occurred
Performing HH comparison with reference data at time =1210675760 ns
Threshold was: -103
Number of pixels with error above threshold: 1048372
Largest error at: range: 1700 azimuth: 276 error: -70
Performing HV comparison with reference data at time =1470732320 ns
Threshold was: -103
Number of pixels with error above threshold: 1048576
Largest error at: range: 466 azimuth: 434 error: -42
Performing VV comparison with reference data at time =1730788800 ns
Threshold was: -103
Number of pixels with error above threshold: 1048359
Largest error at: range: 952 azimuth: 52 error: -130
```

Figure B-5. `testbench.log` *from example Mentor run.*

```
Comparing frame 1 of hvimage.bin to frame 1 of ../../data/m143/C-single/hvimage.bin.
   R  Az          in.r            in.i       err
                ref.i           ref.i
1533  410   148409.468750  -524288.250000  -70.30
            148407.875000 -1048578.000000

 330  291   127254.304688   524287.937500  -76.32
            127254.968750   262144.281250

 937   65   146604.125000   -65536.000000  -88.36
            146604.875000  -131071.062500

 895    1     8044.750000     4096.000000 -112.45
             8045.812500     8191.843750
```

Figure B-6. vh-s.cmp *from example Mentor run.*

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	20 December 1994	Project Report

4. TITLE AND SUBTITLE

RASSP Benchmark 1 Executable Requirements User's Manual

5. FUNDING NUMBERS

C — F19628-95-C-0002
PE — 63739E
PR — 394

6. AUTHOR(S)

Allan H. Anderson, Gary A. Shaw, et al.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lincoln Laboratory, MIT
P.O. Box 73
Lexington, MA 02173-9108

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

ARPA/ESTO
3701 N. Fairfax Dr.
Arlington, VA 22203

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ESC-TR-94-114

11. SUPPLEMENTARY NOTES

None

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This User's Manual describes the installation and use of a set of computer programs and data files which comprise the Executable Requirement for the first benchmark in the RASSP program. The VHDL programs provide a simulation of a Synthetic Aperture Radar processor and a test bench for the simulation. Several C-language utility programs and an image display program are provided. Two sets of data, one from an airborne radar and one synthetic, and images created from the data by a C-language implementation of the SAR algorithm are described. Example scripts for execution of the simulation on Vantage and Mentor simulators are presented along with performance data from example runs.

14. SUBJECT TERMS

RASSP, VHDL, synthetic aperture radar, simulation, Vantage, benchmark, Mentor

15. NUMBER OF PAGES
60

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	